Solutions to ADS practice questions: Introduction to complexity

Question 1.

Given are three different integers a, b and c.

1. Give an algorithm that yields the middle number.
2. How many worst case comparisons will your algorithm need to perform? Can this be improved? How many average case comparisons?

Solution

1. A possible solution:

```
int vindMiddelste(int a, int b, int c) {
  int mid;
  if (a < b) {
    if (b < c)      mid = b;   // a < b < c (1)
    else if (a < c) mid = c;   // a < c < b (2)
    else            mid = a;   // c < a < b (3)
  }
  else if (a < c)   mid = a;   // b < a < c (4)
  else if (b < c)   mid = c;   // b < c < a (5)
  else              mid = b;   // c < b < a (6)

  return mid;
}
```

2. In the worst case (cases 2, 3, 5, 6) the algorithm performs three comparisons. This cannot be improved, as a total ordering of the numbers is required: the information that $a < b$ and $a < c$ is not sufficient to determine whether b or c is the middle number, so the worst case will always be three comparisons.
   The calculation of the average case is based on the assumption that the frequency of each sequence is the same. The worst case then occurs in 4 out of 6 instances and the best case (in which only two comparisons are needed) in 2 out of 6 instances. So the average number of comparisons is

$$\frac{4}{6} \cdot 3 + \frac{2}{6} \cdot 2 = 2\frac{2}{3}.$$

Question 2.

Give an algorithm that yields the largest and smallest element of an array of n numbers; try to give an algorithm that needs about $1.5n$ comparisons in the worst case.

Solution

A simple solution would be to find the minimum and then the maximum. However, this would require $2n - 2$ comparisons. Nor is sorting the array sufficiently efficient; the complexity is $O(n \log n)$. A smarter solution is based on the following strategy:

1. Examine which of the first two numbers in the list is the largest and which is the smallest. Repeat this for the third and fourth numbers and then for every following pair of numbers. Note that one number will remain when n is uneven. This step requires a total of $n/2$ comparisons for an even n, and $(n-1)/2$ when n is uneven.

2. The previous step yields $n/2$ candidates for the largest number when n is even and $(n-1)/2+1$ when n is uneven (the additional candidate is the remaining number). Use a linear search to find the largest element. This requires $n/2-1$ comparisons for an even n, and $(n-1)/2$ when n is uneven.
3. Do the same for all smallest numbers (and once again include the remaining number when n was uneven).

The number of comparisons is now $\frac{n}{2}+\frac{n}{2}-1+\frac{n}{2}-1=\frac{3}{2}n-2$ when n is even, and $\frac{n-1}{2}+\frac{n-1}{2}+\frac{n-1}{2}=\frac{3}{2}(n-1)$ when n is uneven. So in both cases the number of comparisons is about $1.5n$.

An algorithm that combines these steps is as follows (note that this algorithm does not work for empty arrays):

```
(int, int) vindMaxMin (int[] E, int n) {
  int mn, mx;
  if (odd(n)) {
    mn = E[n-1];
    mx = E[n-1];
  }
  else {
    mn = min(E[n-2], E[n-1]);
    mx = max(E[n-2], E[n-1]);
  }
  for (int i = 0; i < n-2; i = i + 2) {
    if (E[i] < E[i+1]) {
      mn = min(mn, E[i]);
      mx = max(mx, E[i+1]);
    }
    else {
      mn = min(mn, E[i+1]);
      mx = max(mx, E[i]);
    }
  }
  return (mn, mx)
}
```

Question 3.

Let $p(n)=a_kn^k+a_{k-1}n^{k-1}+\ldots+a_1n+a_0$ be a polynomial in n with degree k (so $a_k\neq 0$). Proof that $p(n)\in\Theta(n^k)$.

Solution

$$\lim_{n\to\infty}\frac{p(n)}{n^k}=\lim_{n\to\infty}\left(a_k+\frac{a_{k-1}}{n}+\cdots+\frac{a_1}{n^{k-1}}+\frac{a_0}{n^k}\right)=a_k$$

as, evidently, the fractions with n in the denominator and a constant in the numerator approach 0 when n approaches infinity.

As $0<a_k<\infty$, $p(n)\in\Theta(n^k)$ does hold.

Question 4.

Arrange the following functions from the lowest asymptotic order to the highest asymptotic order (and also indicate any functions are of the same asymptotic order):
n, 2^n, $n\log n$, n^3, n^2, $\log n$, $n-n^3+7n^5$, $n^2+\log n$, e^n

(Hint: $\lim_{n\to\infty}\frac{\log n}{n^p}=0$ (for $p>0$) and $\lim_{n\to\infty}\frac{n^p}{a^n}=0$ (for $a>1$).

Solution

$\log n$
n
$n \log n$
n^2, $n^2 + \log n$
n^3
$n - n^3 + 7n^5$
2^n
e^n

The following calculations demonstrate that the asymptotic order does actually increase:

$$\lim_{n\to\infty} \frac{\log n}{n} = 0$$
$$\lim_{n\to\infty} \frac{n}{n \log n} = \lim_{n\to\infty} \frac{1}{\log n} = 0$$
$$\lim_{n\to\infty} \frac{n \log n}{n^2} = \lim_{n\to\infty} \frac{\log n}{n} = 0$$
$$\lim_{n\to\infty} \frac{n^2}{n^3} = \lim_{n\to\infty} \frac{1}{n} = 0$$
$$\lim_{n\to\infty} \frac{n - n^3 + 7n^5}{n^3} = \lim_{n\to\infty} \frac{n}{n^3} - \lim_{n\to\infty} \frac{n^3}{n^3} + \lim_{n\to\infty} \frac{7n^5}{n^3} = 0 - 1 + \infty = \infty$$
$$\lim_{n\to\infty} \frac{n - n^3 + 7n^5}{2^n} = \lim_{n\to\infty} \frac{n}{2^n} - \lim_{n\to\infty} \frac{n^3}{2^n} + \lim_{n\to\infty} \frac{7n^5}{2^n} = 0 - 0 + 0 = 0$$
$$\lim_{n\to\infty} \frac{2^n}{e^n} = \lim_{n\to\infty} \left(\frac{2}{e}\right)^n = \lim_{n\to\infty} (0.7\dots)^n = 0$$

The following demonstrates that n^2 and $n^2 + \log n$ are of the same order:

$$\lim_{n\to\infty} \frac{n^2 + \log n}{n^2} = \lim_{n\to\infty} \frac{n^2}{n^2} + \lim_{n\to\infty} \frac{\log n}{n^2} = 1 + 0 = 1$$

Question 5.

Give the asymptotic order of the solutions to the following recursive formulae, with both recursion trees and the Master theorem. $T(1) = 1$, $n > 1$ and $c > 0$ holds for all comparisons.

1. $T(n) = 2 \cdot T(n/2) + cn$
2. $T(n) = 2 \cdot T(n/2) + cn^2$
3. $T(n) = T(n/2) + cn$

Solution

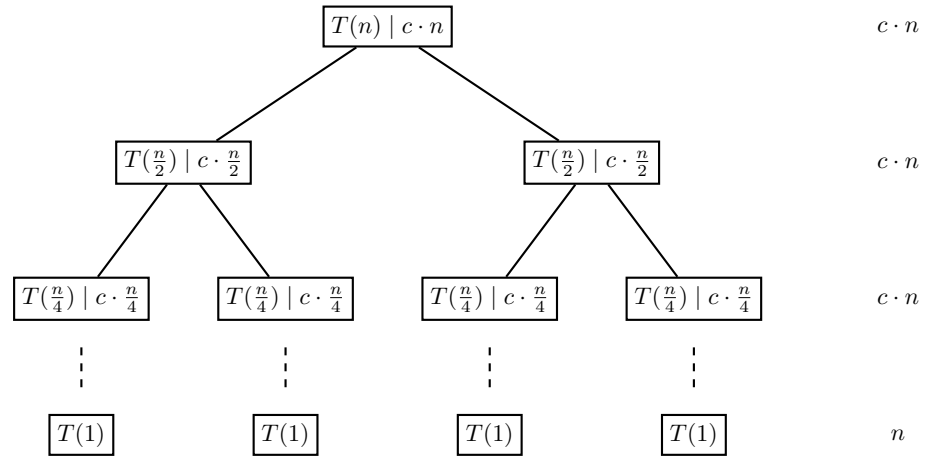
1. *With the Master theorem:*
   In this instance $a = 2$, $b = 2$, and $f(n) = cn$. The following holds

   $$E = {}^b\log a = {}^2\log 2 = \frac{\log 2}{\log 2} = 1.$$

   It then follows directly that $f(n) \in \Theta(n^E)$, so the second case of the Master theorem holds. So,

   $$T(n) \in \Theta(f(n) \cdot \log n) = \Theta(cn \log n) = \Theta(n \log n).$$

With a recursion tree:



There are ${}^2\log n$ levels (excluding the leaves) which each require $c \cdot n$ comparisons, and n leaves which each require 1 comparison. So the total number of comparisons is:

$$T(n) = cn \cdot {}^2\log n + n$$

It then follows directly that $T(n) \in \Theta(n \log n)$.

2. *With the Master theorem:*
In this instance $a = 2$, $b = 2$, and $f(n) = cn^2$. The following holds

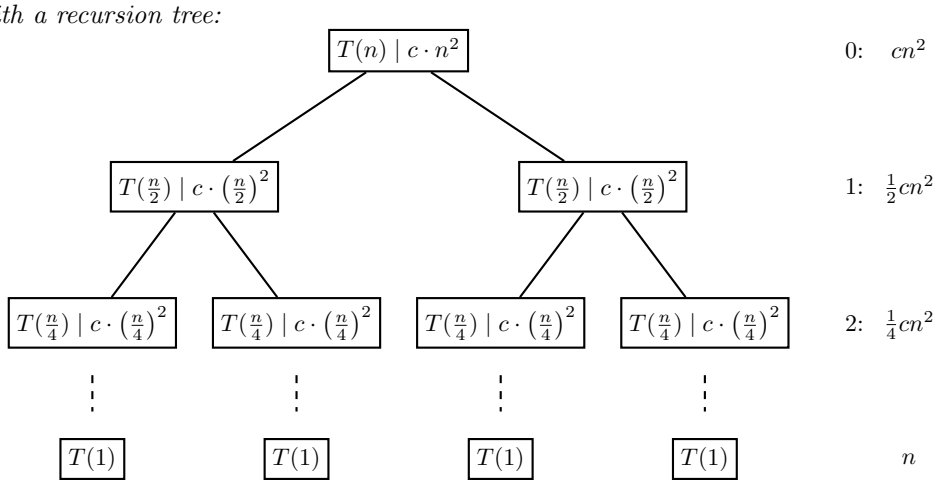
$$E = {}^b\log a = {}^2\log 2 = 1.$$

The first and second case of the Master theorem do not hold, as cn^2 grows more rapidly than $n^E = n$. It does grow rapidly enough for the third case; take, for example, $\epsilon = 0.5$. Then $f(n) \in \Omega(n^{\epsilon+E}) = \Omega(n^{1.5})$ indeed holds.
The regularity condition is also met, as a $0 < c' < 1$ exists such that $af(\frac{n}{b}) \leq c'f(n)$. Take, for example, $c' = \frac{1}{2}$, and then reduce this to

$$af(\frac{n}{b}) = 2c\left(\frac{n}{2}\right)^2 = \frac{1}{2}cn^2 \leq \frac{1}{2}cn^2$$

So, $T(n) \in \Theta(cn^2) = \Theta(n^2)$.

With a recursion tree:



There are ${}^2\log n$ levels (excluding leaves): number these from 0 to ${}^2\log n - 1$, as shown next to the figure. Now at level i the number of comparisons required is $\left(\frac{1}{2}\right)^i cn^2$. There are also n leaves, which each require 1 comparison. So the total number of comparisons is:

$$\begin{aligned}
T(n) &= n + \sum_{i=0}^{({}^2\log n)-1} \left(\frac{1}{2}\right)^i cn^2 \\
&= n + cn^2 \cdot \sum_{i=0}^{({}^2\log n)-1} \left(\frac{1}{2}\right)^i \\
&= n + cn^2 \cdot \frac{1 - \left(\frac{1}{2}\right)^{{}^2\log n}}{1 - \frac{1}{2}} \\
&= n + 2cn^2 \cdot \left(1 - n^{{}^2\log \frac{1}{2}}\right) \\
&= n + 2cn^2 \cdot \left(1 - n^{-1}\right) \\
&= n + 2cn^2 - 2cn
\end{aligned}$$

From this it follows that $T(n) \in \Theta(n^2)$.

3. *With the Master theorem:*
In this instance $a = 1$, $b = 2$ and $f(n) = cn$. The following holds

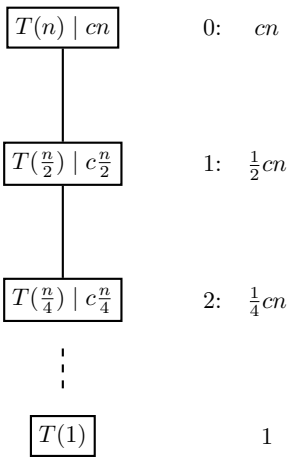
$$E = {}^b\log a = {}^2\log 1 = 0.$$

It is clear that the first and second case of the Master theorem do not hold, as $f(n)$ grows faster than $n^E = 1$. It does grow rapidly enough for the third case; take, for example, $\epsilon = 0.5$. Then $f(n) \in \Omega(n^{\epsilon+E}) = \Omega(n^{0.5})$ does indeed hold.
The regularity condition is also met, as an $0 < c' < 1$ exists such that $af(\frac{n}{b}) \leq c'f(n)$. Take, for example, $c' = \frac{1}{2}$, and then reduce this to

$$af(\frac{n}{b}) = 1c\left(\frac{n}{2}\right) = \frac{1}{2}cn \leq \frac{1}{2}cn$$

So, $T(n) \in \Theta(cn) = \Theta(n)$.

With a recursion tree:



There are ${}^2\log n$ levels (excluding leaves): number these from 0 tot ${}^2\log n - 1$, as shown next to the figure. Now at level i the number of comparisons required is $\left(\frac{1}{2}\right)^i cn$. There is also 1 leaf, which requires 1 comparison. So the total number of comparisons is:

$$\begin{aligned}
T(n) &= 1 + \sum_{i=0}^{({}^2\log n)-1} \left(\frac{1}{2}\right)^i cn \\
&= 1 + cn \cdot \sum_{i=0}^{({}^2\log n)-1} \left(\frac{1}{2}\right)^i \\
&= 1 + cn \cdot \frac{1 - \left(\frac{1}{2}\right)^{{}^2\log n}}{1 - \frac{1}{2}} \\
&= 1 + 2cn \cdot \left(1 - n^{{}^2\log \frac{1}{2}}\right) \\
&= 1 + 2cn \cdot \left(1 - n^{-1}\right) \\
&= 1 + 2cn - 2c
\end{aligned}$$

From this it follows that $T(n) \in \Theta(n)$.

Question 6.

The Tower of Hanoi: a number of discs of different sizes are stacked on a peg (the *start* peg), in order of size (with the largest disc at the bottom). There are two empty pegs next to this first peg, the *spare* and *destination* pegs. The objective is to move the entire stack from the *start* to the *destination*, disc by disc, without placing any disc on top of a smaller disc. The *spare* peg may be used when moving the discs. A recursive solution:

```
void hanoi(numberOfDisks, start, destination, spare)
{  if (numberOfDisks > 0) {
       hanoi(numberOfDisks - 1, start, spare, destination);
       move top disk from peg start to peg destination;
       hanoi(numerOfDisks - 1, spare, destination, start); }
}
```

Give a recursive formula for the number of moves and solve the equation.

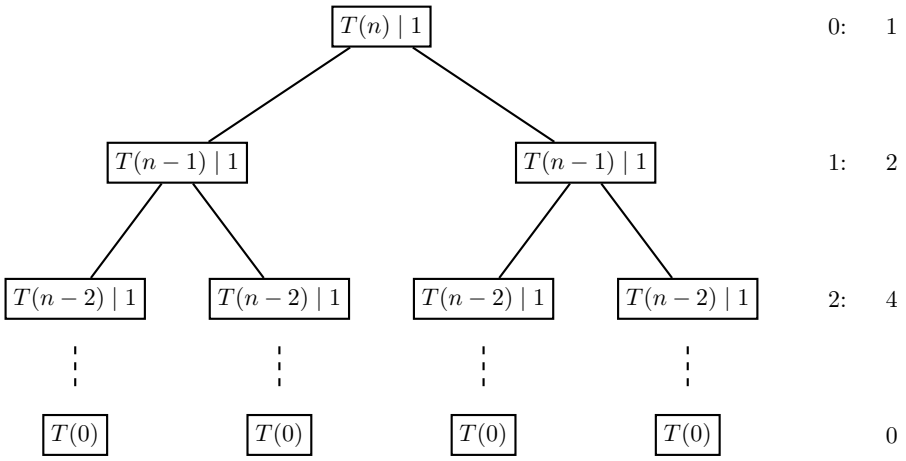
Solution

The input size is the number of discs used in the puzzle. The basic operation is the moving of a disc. The recursive equation is:

$$\begin{aligned}
&T(n) = 2T(n-1) + 1 \qquad (n > 0) \\
&T(0) = 0
\end{aligned}$$

This is because two recursive calls are made with an input that is 1 smaller. Moreover, only 1 iteration is done to 'link' these two iterations.

As the recursive formula is not in the correct format for the use of the Master theorem, use a recursion tree.



There are n levels (excluding leaves): number these from 0 to $n-1$, as shown next to the figure. Now at level i the number of comparisons required is equal to 2^i. As there are no comparisons for the leaves, these do not need to be taken into account. So the total number of comparisons is:

$$T(n) = \sum_{i=0}^{n-1} 2^i = \frac{1-2^n}{1-2} = \frac{2^n-1}{2-1} = 2^n - 1$$

From this it follows that $T(n) \in \Theta(2^n)$.